Amendments to the Claims

Claim 1 (currently amended): A method for programmatically enforcing referential integrity constraints among associations between class instances, comprising steps of:

programmatically determining, when evaluating a request to modify an existing association end of a bi-directional link to reflect an association from an instance of a first class to an instance of a second class, whether the association end to be modified has a single multiplicity or a many multiplicity;

if the association end to be modified has the single multiplicity, modifying the existing association end by atomically and programmatically performing the steps of:

first disconnecting a previously-existing inverse association end of the requested association end;

next setting an inverse association end of the association to reflect an inverse association from the instance of the second class to the instance of the first class; and

then setting the requested association end from the instance of the first class to the instance of the second class; and

if the association end to be modified has the many multiplicity, modifying the existing association end by atomically and programmatically performing the steps of:

first adding the requested association end from the instance of the ~~second~~ first class to the instance of the [[first]] second class;

next disconnecting the previously-existing inverse association end of the requested association end; and

then setting an inverse association end of the association to reflect an inverse

association from the instance of the [[first]] second class to the instance of the ~~second~~ first class.

Claims 2 - 4 (canceled)

Claim 5 (previously presented): The method according to Claim 1, further comprising the step of serializing the association by performing steps of:

determining whether the association end to be modified or the inverse association end is a primary end of the association; and

serializing only the primary end of the association during the serialization.

Claim 6 (previously presented): The method according to Claim 1, wherein the method is provided as a single link helper object and a multiple link helper object for each association, wherein the single link helper object performs the atomically and programmatically performed steps for the single multiplicity association end and the multiple link helper object performs the atomically and programmatically performed steps for the many multiplicity association end.

Claim 7 (currently amended): A computer program product for programmatically enforcing referential integrity constraints among associations between class instances, wherein the computer program product is embodied on one or more computer readable media and comprises computer-readable program code means for:

programmatically determining, when evaluating a request to modify an existing association end of a bi-directional link to reflect an association from an instance of a first class to

an instance of a second class, whether the association end to be modified has a single multiplicity or a many multiplicity;

if the association end to be modified has the single multiplicity, modifying the existing association end by atomically and programmatically performing the steps of:

first disconnecting a previously-existing inverse association end of the requested association end;

next setting an inverse association end of the association to reflect an inverse association from the instance of the second class to the instance of the first class; and

then setting the requested association end from the instance of the first class to the instance of the second class; and

if the association end to be modified has the many multiplicity, modifying the existing association end by atomically and programmatically performing the steps of:

first adding the requested association end from the instance of the ~~second~~ first class to the instance of the [[first]] second class;

next disconnecting the previously-existing inverse association end of the requested association end; and

then setting an inverse association end of the association to reflect an inverse association from the instance of the [[first]] second class to the instance of the ~~second~~ first class[[,]].

Claims 8 - 9 (canceled)

Claim 10 (previously presented): The computer program product according to Claim 7, further comprising computer-readable program code means for serializing the association by performing steps of:

determining whether the association end to be modified or the inverse association end is a primary end of the association; and

computer-readable program code means for serializing only the primary end of the association during the serialization.

Claim 11 (currently amended): A system for programmatically enforcing referential integrity constraints among associations between class instances, comprising means for:

programmatically determining, when evaluating a request to modify an existing association end of a bi-directional link to reflect an association from an instance of a first class to an instance of a second class, whether the association end to be modified has a single multiplicity or a many multiplicity;

if the association end to be modified has the single multiplicity, modifying the existing association end by atomically and programmatically performing the steps of:

first disconnecting a previously-existing inverse association end of the requested association end;

next setting an inverse association end of the association to reflect an inverse association from the instance of the second class to the instance of the first class; and

then setting the requested association end from the instance of the first class to the instance of the second class; and

if the association end to be modified has the many multiplicity, modifying the existing association end by atomically and programmatically performing the steps of:

first adding the requested association end from the instance of the ~~second~~ first class to the instance of the [[first]] second class;

next disconnecting the previously-existing inverse association end of the requested association end; and

then setting an inverse association end of the association to reflect an inverse association from the instance of the [[first]] second class to the instance of the ~~second~~ first class.

Claims 12 - 13 (canceled)

Claim 14 (previously presented): The system according to Claim 11, further comprising means for serializing the association by performing steps of:

determining whether the association end to be modified or the inverse association end is a primary end of the association; and

means for serializing only the primary end of the association during the serialization.

Claim 15 (previously presented): The method according to Claim 1, wherein one or more structured markup language representations specify instances of the first class, instances of the second class, and associations between the instances of the first and second classes.

Claim 16 (previously presented): The method according to Claim 15, wherein only one

association end for each association between instances is specified in the structured markup language representations.

Claim 17 (previously presented): The method according to Claim 16, wherein the only one association end is an association end designated as a primary end for the association.

Claim 18 (previously presented): The method according to Claim 15, wherein a serialization of results of the request to modify the existing association end that has the single multiplicity comprises the step of:

determining whether the association end to be modified is a primary end for the association, and if so, programmatically performing the steps of:

removing the representation of the previously-existing inverse association end from the structured markup language representation in which it is specified; and

adding a structured markup language representation of the new inverse association end.